RC23056 (W0404-055) April 8, 2004
Computer Science

IBM Research Report

IBM Research and the University of Colorado TREC 2003 Genomics Track

Eric W. Brown
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Andrew Dolbey, Lawrence Hunter
School of Medicine
University of Colorado
Denver, CO 80262

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

IBM Research and the University of Colorado TREC 2003 Genomics Track

Eric W. Brown[*], Andrew Dolbey[†], Lawrence Hunter[†]

[*]IBM TJ Watson Research Center
PO Box 704
Yorktown Heights, NY 10598
ewb@us.ibm.com

[†]School of Medicine
University of Colorado
Denver, CO 80262
{Andrew.Dolbey, Larry.Hunter}@uchsc.edu

Introduction

IBM Research and the University of Colorado collaborated on their submission to the inaugural Genomics track at TREC 2003. IBM Research has extensive experience in natural language processing, text analysis, and large-scale systems [9, 13, 3, 5, 16, 10]. IBM also has numerous research and business activities in the broad areas of bioinformatics and bio-medical information processing [14, 8]. IBM Research is currently developing *BioTeKS*, a middleware system for text analysis, mining, and information retrieval in the bio-medical domain. The University of Colorado (CU) has been working in the area of bioinformatics and text analysis in the bio-medical domain for a number of years and has made substantial contributions to the field [7, 11, 15, 12]. CU contributed their domain expertise to enhance the BioTeKS system and jointly we designed and evaluated experiments while preparing our track submissions.

The basic premise of BioTeKS is that the best way to enable effective exploitation of vast text resources is to associate meaningful semantics with the tokens and phrases in the text. With a better understanding of the semantic content of text as a foundation, we can build information extraction, summarization, and indexing systems that address specific information needs in complex domains. For example, bio-medical researchers often need to find documents that contain specific entities (e.g., genes, proteins, cellular components) interacting in certain ways. To satisfy such requests, we must first be able to identify the entities (which may be named using a variety of aliases or synonyms) and then recognize textual constructs that describe these entities interacting with the desired relationships.

BioTeKS is built on the IBM Unstructured Information Management Architecture (UIMA) [6], which is a framework for building unstructured information analysis applications. UIMA provides a number of standard facilities for managing the flow of data through the system, scheduling and orchestrating low-level analysis tasks, and assembling, analyzing, and storing results. BioTeKS uses the UIMA framework to assemble text analysis engines that provide tokenization, named entity recognition, part of speech tagging, shallow and deep parsing, relationship extraction, and semantic indexing.

For our Genomics track submissions, we focused on developing named entity recognizers for genes, proteins, and functions. Our basic recognizer was dictionary based, where each dictionary entry contained all known synonyms for the corresponding entity, and matching of synonyms against the text involved normalization heuristics appropriate for the entity type. For example, authors are often lax in their use of capitalization, spaces, hyphens, and slashes when writing gene symbols [4]. Our matching heuristics consider this behavior, and much of our pre-submission work involved experiments to determine which heuristics provide the best balance of precision and recall for the Genomics track tasks. Our dictionary of gene and protein names was derived from the full LocusLink database, and our function dictionary was derived based on a statistical analysis of verbs and related nominalizations that frequently co-occur with the gene of interest in the Genomics track training data.

For the primary task (ad-hoc retrieval), we analyzed the test corpus with our named entity recognizers and created annotations in the text for recognized entities. An annotation spans the original text in the document and contains meta-data about the annotation, such as a canonical form and the semantic role of the entity. We then used the JuruXML search engine [1, 2] to index the full text and annotations for each document in the corpus. The JuruXML query language supports both free-text queries as well as queries over annotations, annotation attributes, and the text spanned by annotations. We automatically generated the JuruXML queries from the test topics, with the final query generation algorithm selected based on experiments with the training data. Our final average precision over 11 points of recall for the 50 test queries was 0.28.

For the secondary task (information extraction/summarization), we applied the same set of named entity recognizers to annotate the text documents. We then scored each sentence using a weighted combination of features, including annotations, location in the document, and structural role of the sentence (e.g., title). The weights were determined empirically, and the best scoring sentence was returned as the summary.

In the remaining sections we describe our overall architecture, present our approach in more detail, briefly analyze our results, and close with conclusions.

Architecture

Our BioTeKS system is built on the IBM Unstructured Information Management Architecture (UIMA) [6], which is summarized in Figure 1. The UIMA provides a framework for implementing and deploying an unstructured information processing and analysis system. Unstructured information (text in this scenario) is fed to an Application Logic layer, which represents a specific instantiation of the architecture for the current domain and provides an application level interface to the framework. The Application Logic layer passes documents to the Collection Analysis Engine, which calls on the Collection Processing Manager to orchestrate the text analysis processing steps. The actual text analysis operations are performed by pluggable Text Analysis Engines.

A Text Analysis Engine, or TAE, performs a specific text analysis task, such as tokenization, lemmatization, part of speech tagging, parsing, named entity recognition, relationship extraction, etc. TAEs may operate directly on document content, or they may process the output of previously run TAEs. The UIMA framework defines a standard API for building a TAE and describing its functionality, inputs, and outputs. For TAEs that require access to other information resources during analysis, the Structured Knowledge Access module provides mechanisms for accessing various knowledge resources, such as dictionaries, lexicons, or ontologies. These resources may be provided locally, or they may be standard, external resources (e.g., MeSH, UMLS, GO) with appropriate Knowledge Source Adapters that allow access to the resource through the framework.

When the text analysis processing steps for a given document are complete, the Collection Processing Manager submits the results for indexing by the Semantic Search Engine (JuruXML), stores selected analysis results and document meta-data in the Store, and returns the results to the Collection Analysis Engine. The Collection Analysis Engine accumulates results over all of the documents in the collection and performs any collection-wide analyses specified by the application logic layer, saving those results in the Store.

The UIMA framework exploits standard middleware software to implement various components of the framework as appropriate (e.g., a relational database management system, such as IBM DB2™, for the Store, or an application server, such as IBM WebSphere™, for deploying Text Analysis Engines as web services). UIMA in turn provides support for deploying Collection Processing and Text Analysis steps in a variety of local, distributed, and parallel configurations, depending on the underlying computing infrastructure.

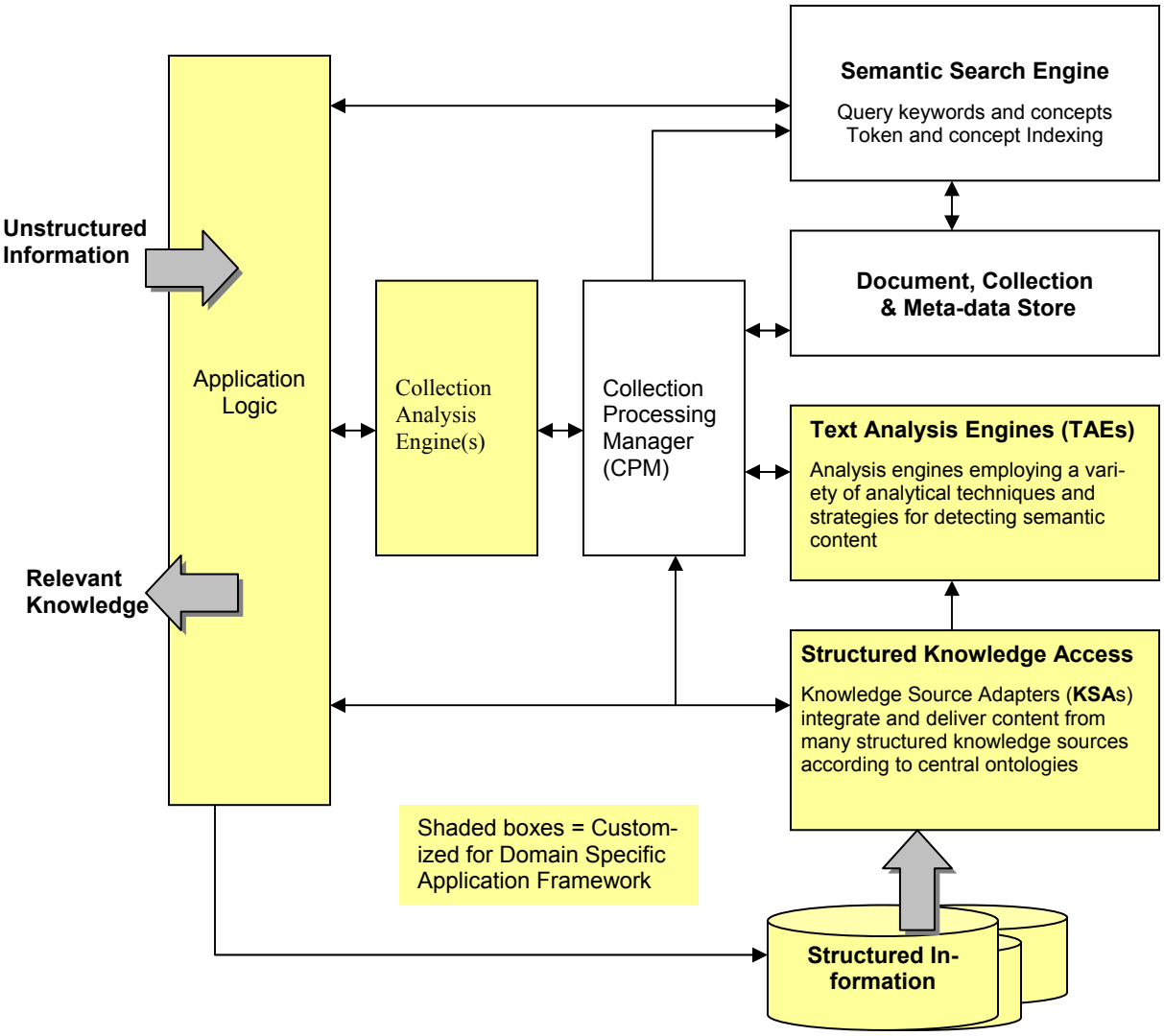


Figure 1 UIMA High-Level Architecture

When all of the documents have been processed, the Application Logic layer provides access to the processing results either via the Semantic Search Engine or through custom access functions to the Store. The Semantic Search Engine is a key component of the system, providing the ability to express complex and sophisticated queries over both the raw text in the documents and the results of the text analysis processing.

Approach

Task 1

Our overall approach to solving the Task 1 problem (retrieve documents that describe the function of a given gene) was to parse and tokenize the MEDLINE abstracts, recognize gene mentions and annotate them with a canonical form, recognize "function words" that are indicative of gene function and annotate them with a canonical form, index the full text plus annotations with JuruXML, and automatically generate JuruXML queries. The text analysis flow is summarized in Figure 2.

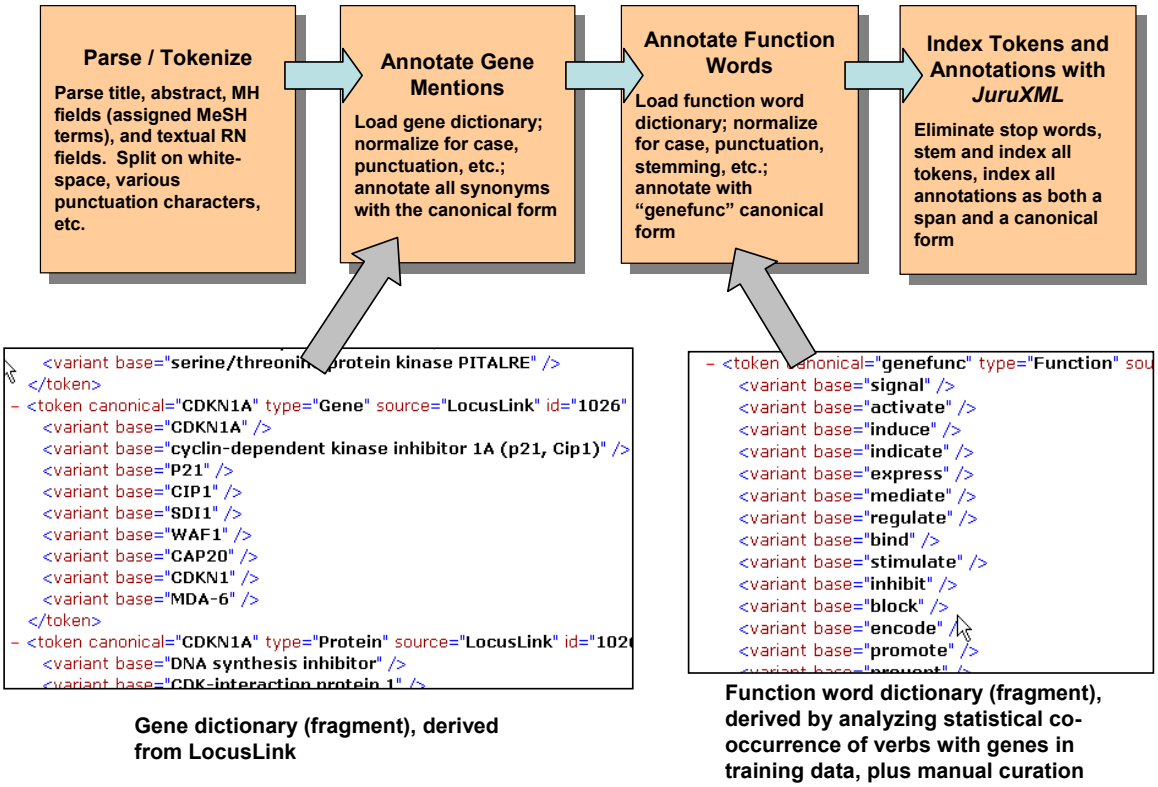


Figure 2 BioTeKS processing steps and dictionaries

To recognize gene mentions and function words, we used a dictionary-based named entity recognizer implemented as a UIMA Text Analysis Engine. The dictionary contains an entry for each named entity, which in turn includes all known synonyms for the entity, a "canonical" or preferred name for the entity, and additional optional meta-data associated with the entity. A synonym may be a single token or a multi-token phrase. The entity recognizer TAE scans the input text and at each token searches for the longest matching synonym in the dictionary. When a matching synonym is found an annotation is created in the text that spans the matching tokens. The annotation includes the canonical form for the entity and all other meta-data specified in the dictionary entry for the entity. The TAE will optionally perform stemming and case folding when attempting to match the text against the dictionary of synonyms, and the set of characters used to separate tokens is configurable.

The dictionary for finding gene mentions was automatically derived from the full LocusLink database, and included 156,533 genes with a total of 387,850 synonyms. The preferred gene symbol was used for the canonical form and the synonyms were extracted from the LocusLink entry fields that contain the known gene or protein aliases used for the gene. During dictionary matching we did not use stemming, but we did case fold all tokens that contained at least one numeric character, and the set of characters used to separate tokens included white space, punctuation characters, and in particular hyphen, forward and backward slash, and parentheses. Tokenizing on these characters and eliminating them from the tokens improved the recall of our gene identifier and addressed some of the variability found in gene names associated with inconsistent use of space, hyphen, slashes, and parentheses.

The dictionary of function words was derived in a semi-automatic fashion. Using the training data for Task 1, we identified verbs that frequently occur in sentences with genes as the subject. We sorted this list based on a scoring function of the significance of this co-occurrence, and then manually curated the list to select important gene function words, yielding a rather small list of 28 function words. We used

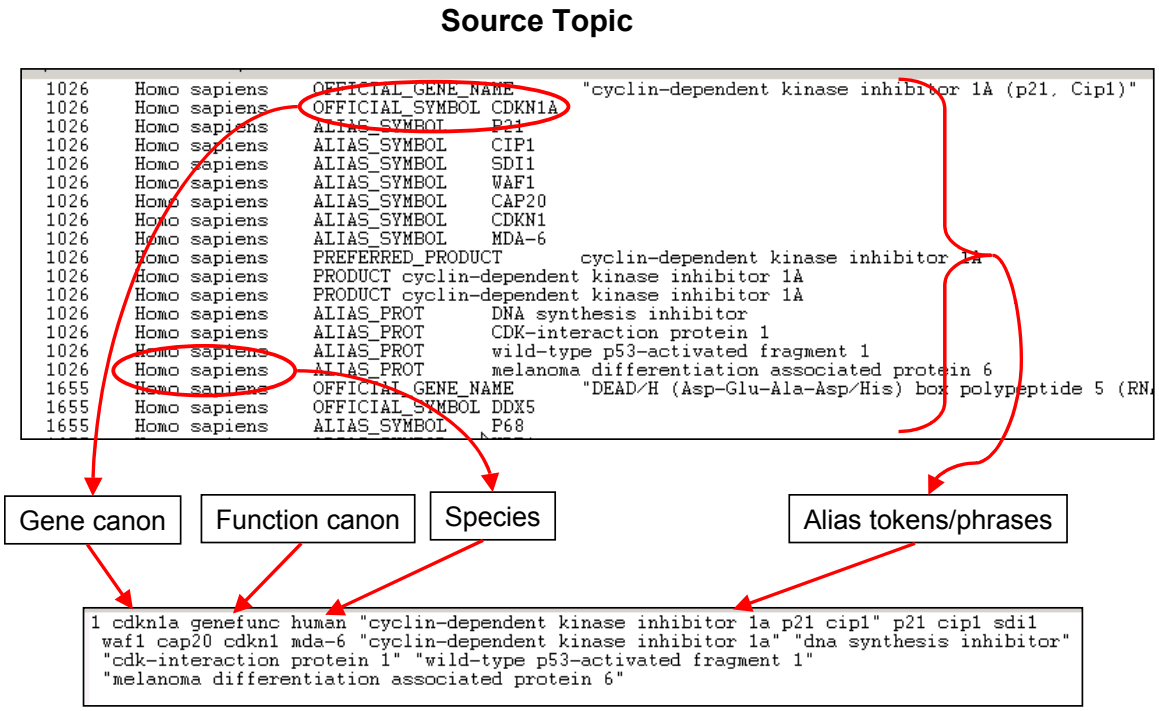


Figure 3 Task 1 automatic query query generation.

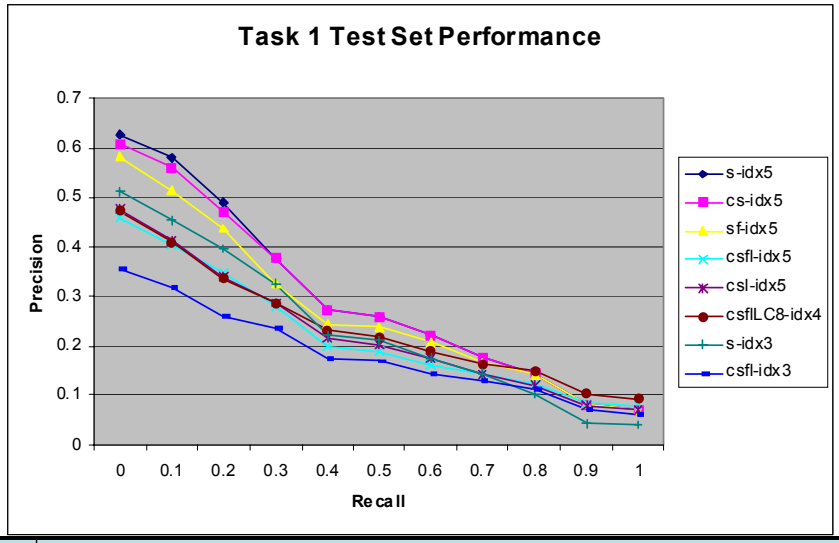
stemming and case folding during matching, allowing the function word finder to match noun forms as well as verb forms. Samples of our dictionaries are shown in Figure 2.

For the Task 1 queries, we explored using the gene canon, the species name (*human*, *mice* / *mouse*, *rat*, or *Drosophila*), the function word canon, and all given gene aliases as tokens or phrases. We automatically generated queries for JuruXML based on the provided query source topics. Our query generation process is summarized in Figure 3.

Task 2

For Task 2 (automatically extract and summarize a gene's function given a document known to describe the gene's function), we decided to chunk the document into whole sentences, score each sentence, and return the best scoring sentence as our answer. A sentence's score is based on whether or not it contains the target gene, how many gene function words it contains, what structural role the sentence plays (i.e., is it the *title*), and where in the document the sentence occurs. To identify genes and function words in the documents we applied the text analysis processing steps shown in Figure 2, excluding the final step of indexing with JuruXML.

Given that we had committed to extracting the single best scoring sentence as the summary, we performed a simple analysis to determine if it was worthwhile to analyze the full article versus just the MEDLINE abstract. For each document we scored every sentence in the document against the gold standard (the GeneRIF for the given gene and article) using the classic Dice coefficient (as implemented in the scoring code provided by the track) and identified the best scoring sentence. We then returned this sentence as our answer and calculated the average performance over the set of documents. This essentially produces an upper bound the best possible score that could be obtained assuming a strategy of returning the single best sentence.



Key	Query Generation Option (Query always includes canonical gene name)
s	Species name included in query
c	For gene name "X" ambiguous w/ common words, "X" goes to "X gene" and "X protein"
f	Gene function annotation term used in query
l	Long form: all topic fields used, multi-word fields in quotes (phrases), all tokens & phrases unique
lL	Same as long form, but multi-word fields are not in quotes
Cn	Canonical gene name (official or preferred name) repeated n times
	Text Analysis / Indexing Option
idx4,5	Gene matching done with case-folding of names with 1+ digits, hyphens stripped
idx3	Heuristic expansion of hyphen/space combinations

Figure 4 Task 1 test set performance.

For the set of full-length articles, the optimal classic Dice score is 70.61%. For the set of MEDLINE abstracts, the optimal classic Dice score is 71.09%. This result is somewhat surprising given that the abstract should be a proper subset of the full-length article. This anomaly is due to the following: most GeneRIFs are extracted directly from the MEDLINE abstracts. The full-length article contains SGML entities that must be translated to ASCII for the MEDLINE abstract, e.g., '&agr;' -> 'alpha'. This translation is not done consistently, such that a fragment extracted from the full text may not exactly match the GeneRIF using the Dice measures. Given this result, we chose to use the abstracts rather than the full-length articles for our actual Task 2 run.

Results

Task 1

Using the training queries for Task 1 we explored a variety of query generation options and measured the performance of the system. On the training queries we were able to obtain an 11pt average precision of 0.4259. Based on these results, we submitted two runs on the test queries. Run **IBMbt1** was generated using queries that comprise only the gene canonical form and the species name. This run produced an 11pt average precision of 0.2823, but found only 456 of 566 possible relevant documents. Run **IBMbt2** was generated using queries that comprise the gene canonical form, species name, function keyword, and all alias forms from the source query topic. This run produced an 11pt average precision of 0.2259 while returning 534 of 566 possible relevant documents. Adding more terms to the query improved recall but resulted in poorer overall ranking.

With the relevance judgments for the test queries we performed a more detailed analysis of various query generation options on the test data. These results are shown in Figure 4. From the plot we see that the relatively simple query **s-idx5** (species name and gene canonical form) produces better precision at low recall, while the queries with additional terms produce worse precision at higher recall levels.

Task 2

Although we explored a variety of parameter settings for our sentence scoring function, we were not able to obtain a scoring function that performed better than simply returning the title. We are currently exploring a number of ways to improve our scoring function, such as incorporating a shallow parse in the analysis to more accurately connect the target gene with the function words in the sentence.

Conclusion

Based on our results, we conclude that using a comprehensive gene dictionary with appropriate normalization during matching is an effective way to annotate gene mentions in biomedical text. The normalization and matching heuristics are very important, however, given the considerable variability found in gene names, especially in the use of capitalization, spaces, hyphens, slashes, and parenthesis. Unfortunately, identifying gene "function words" is not necessarily useful in a bag-of-words query context. We suspect, however, that they might be more effective when identified in syntactic relationships with genes. This will require the addition of parsing (e.g., shallow parsing) to the analysis phase.

The significant difference in our training and test results for Task 1 (a phenomenon observed by many of the Task 1 participants) suggests that the overall test set is not stable. There may be too few relevant documents for some of the test queries, or the relevance judgments may be too incomplete. This latter issue is particularly important and was raised by a number of the track participants.

Given the exploratory nature of Task 2 and the relatively late decision by the track to collect official runs for the task, we did not invest as much time in this Task. In the process of developing our sentence scoring function, we observed a number of cases where our extracted sentence appeared to convey the same meaning as the gold standard, but due to the wording the sentence scored poorly using the various Dice measures. Based on our experience and the experience of others on this task, we are not convinced that this particular evaluation accurately measures a system's ability to perform what is arguably an important real world task.

Given the overall constraints under which this inaugural Genomics track was run, we feel the track was very successful and accomplished its goals for the first year. In particular, it brought this important area of research to the attention of the Information Retrieval community and made a positive step in the direction of building useful test sets in this domain, which currently suffers from a severe lack of well constructed test sets. We look forward to next year's track and the development of more realistic tasks supported by more thorough evaluation.

References

[1] D. Carmel, E. Amitay, M. Herscovici, Y. S. Maarek, Y. Petruschka and A. Soffer, Juru at TREC 10--Experiments with Index Pruning, *Proceedings of The Tenth Text REtrieval Conference (TREC 2001)*, 2002,

[2] D. Carmel, Y. S. Maarek, M. Mandelbrod, Y. Mass and A. Soffer, Searching XML Documents via XML Fragments, *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, Toronto, Canada, 2003, 151-158.

[3] J. Chu-Carroll, J. Prager, C. Welty, K. Czuba and D. Ferrucci, A Multi-Strategy and Multi-Source Approach to Question Answering, *Proceedings of The Eleventh Text REtrieval Conference (TREC 2002)*, Gaithersburg, MD, 2003,

[4] K. B. Cohen, A. Dolbey, G. Acquaah-Mensah and L. Hunter, Contrast and Variability in Gene Names, *Proceedings of the ACL Workshop on Natural Language Processing in the Biomedical Domain*, Philadelphia, PA, 2002, 14-20.

[5] J. W. Cooper and R. J. Byrd, Lexical Navigation: Visually Prompted Query Expansion and Refinement, *Proceedings of the ACM International Conference on Digital Libraries*, Philadelphia, PA, 1997, 237-246.

[6] D. Ferrucci and A. Lally, Accelerating Corporate Research in the Development, Application and Deployment of Human Language Technologies, *Proceedings of the HLT-NAACL 2003 Workshop on Software Engineering and Architectures for Language Technology Systems*, Edmonton, Canada, 2003, 68-75.

[7] L. Hunter, R. C. Taylor, S. M. Leach and R. Simon, GEST: a gene expression search tool based on a novel Bayesian similarity metric, Bioinformatics, 17 (2001), pp. S115-S122.

[8] IBM, Life Sciences Solutions, http://www-3.ibm.com/solutions/lifesciences/.

[9] A. Ittycheriah and S. Roukos, IBM's Statistical Question Answering System-TREC 11, *Proceedings of The Eleventh Text REtrieval Conference (TREC 2002)*, Gaithersburg, MD, 2003,

[10] D. E. Johnson, F. J. Oles, T. Zhang and T. Goetz, A decision-tree-based symbolic rule induction system for text categorization, IBM Systems Journal, 41 (2002), pp. 428-437.

[11] P. V. Ogren, K. B. Cohen, G. Acquaah-Mensah, J. Eberlein and L. Hunter, The Compositional Structure of Gene Ontology Terms, *Proceedings of the Pacific Symposium on Biocomputing*, 2004, 9:214-225.

[12] T. Phang, M. Rudolph, M. Neville and L. Hunter, Trajectory Clustering: A Non-Parametric Method for Grouping Gene Expression Time Courses, With Applications to Mammary Development, *Proceedings of Pacific Symposium on Biocomputing*, 2003, 8:351-362.

[13] J. Prager, E. Brown, A. Coden and D. Raden, Question-Answering by Predictive Annotation, *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Athens, Greece, 2000, 184-191.

[14] I. Rigoutsos, T. Huynh, A. Floratos, L. Parida and D. Platt, Dictionary-driven Protein Annotation, Nucleic Acids Research, 30 (2002).

[15] R. Shenkar, J. P. Elliott, K. Diener, J. Gault, L. J. Hu, R. J. Cohrs, T. Phang, L. Hunter, R. E. Breeze and I. A. Awad, Differential gene expression in human cerebrovascular malformations, Neurosurgery, 52 (2003), pp. 465-477.

[16] T. Zhang, F. Damerau and D. Johnson, Text Chunking based on a Generalization of Winnow, Journal of Machine Learning Research, 2 (2002), pp. 615-637.